Redes Neuronales y Árboles de Decisión: Un Enfoque Híbrido

Goddard J.C.; Cornejo J.M.; Martínez F.M.; Martínez A.E. [(#)]
Rufiner H.L.; Acevedo R.C. [(#)] [(*)]
[(#)] *Laboratorio de Audiología, Departamento de Ingeniería Eléctrica*
Universidad Autónoma Metropolitana - Iztapalapa
[(*)] *Universidad Nacional de Entre Ríos, Argentina*.

Resumen: Actualmente dos de los métodos de aprendizaje maquinal más ampliamente utilizados en tareas de clasificación son los árboles de decisión (AD) y las redes neuronales (RN). En este último caso la arquitectura más utilizada son los perceptrones multicapa (PMC). Sin embargo, cada uno de ellos puede presentar distintas dificultades en las aplicaciones del mundo real. Por ejemplo, los AD pueden ser pocos flexibles para generalizar sobre datos de prueba y excesivamente ramificados. En el caso de los PMC hay que definir su estructura -número de nodos y capas - y aún definida esta no hay garantía que converja a una solución aceptable. En el presente trabajo se describe un método para implementar un PMC a partir de un AD. Posteriormente se compara el desempeño de este PMC en relación al AD original y con respeto a un PMC definido por separado. Para la comparación se utiliza la conocida base de datos IRIS, que ha sido usada de manera universal en la literatura de aprendizaje maquinal.

I. INTRODUCCIÓN

La clasificación mediante AD es un método no paramétrico ampliamente utilizado en el reconocimiento de patrones complejos, especialmente cuando en la tarea de clasificación se involucran varias clases de patrones y un gran número de atributos. En este método se emplean funciones de decisión, generalmente binarias, a fin de determinar la identidad de un patrón desconocido. El propósito que se persigue es que en cada evaluación sucesiva de una función de decisión se reduzca la incertidumbre en la identificación del patrón desconocido. En la mayoría de los casos el argumento de cada una de estas funciones es un elemento del conjunto de atributos. Esto, junto con el hecho de que patrones de diferentes clases se identifican utilizando diferentes conjuntos de funciones de decisión, hacen que la clasificación mediante un AD sea computacionalmente atractiva. En cierto sentido, la clasificación mediante un AD es la forma "óptima" de comparar los atributos con los subconjuntos de clases de patrones en cada una de las etapas intermedias en una tarea de clasificación.

En AD binarios se comienza con la función de decisión del nodo raíz, los nodos internos particionan sucesivamente la región de decisión en dos espacios, donde la frontera que los divide está definida por la función del nodo correspondiente. De esta manera se logra una partición jerarquizada del espacio de decisiones. Dependiendo de las características de la función del nodo y del tamaño del árbol, la frontera final de decisión puede ser muy complicada. Una de las funciones más empleadas es la prueba con un cierto umbral para cada componente del vector de atributos, teniendo como resultado la partición del espacio de atributos por medio de hiperplanos paralelos u ortogonales a los ejes coordenados del espacio de atributos. Como característica importante del AD se puede mencionar que a partir de una colección de vectores de patrones etiquetados se configura de manera autónoma, independientemente de cualquier información *a priori* acerca de la distribución de los vectores de patrones en el espacio de decisiones. En lo que concierne con el aprendizaje de máquina, la construcción de AD es sinónimo de la adquisi-

ción de conocimiento estructurado en forma de conceptos y reglas para sistemas expertos.

Aunque los AD son intuitivamente atractivos y han tenido aplicaciones exitosas, [Qui93], [BFO84] existen algunos problemas que pueden obstaculizar su empleo en casos reales. Estos problemas incluyen rubros tales como el diseño del árbol en si mismo, la presencia de datos inconclusos, incompletos o ruidosos y el empleo simultáneo de todos los vectores de atributos.

Para solucionar algunos de estos problemas se debe pensar en algún criterio adecuado de terminación, es decir, hasta que punto se debe hacer crecer el árbol de tal forma que no se obtenga un árbol demasiado complejo, difícil de entender y poco eficiente.

Es conveniente mencionar que hay algunos algoritmos que incluyen algún criterio de podado del árbol, tal es el caso de *CART* [BFO84]. Aquí la partición recursiva continúa hasta que el árbol comienza a ser muy grande, en ese momento se comienza a podar el árbol hacia arriba hasta encontrar un mejor subárbol tratando de obtener el más bajo error estimado. El incluir la técnica de podado a estos algoritmos implica pagar un precio extra que generalmente esta dado por un aumento en el gasto computacional y/o una disminución en la precisión. Otras posibles soluciones se plantean con *C4* [Qui93] y una extensión a *CART* en [GRD91].

Algunos otros problemas han sido atacados en el pasado por los investigadores dentro del marco del reconocimiento estadístico de patrones y el aprendizaje maquinal; lo que ha llevado a plantear que el desempeño de los árboles clasificadores puede mejorarse explotando algunas de las características claves de los PMC. Esto se debe a que existen similitudes, en ciertos aspectos, con los árboles de clasificación [Set91],[SeO90]. Por ejemplo, ambos tipos de clasificadores no imponen restricción alguna sobre la distribución de las observaciones de entrada y a partir de un conjunto de vectores de entrenamiento pueden generar fronteras de decisión arbitrariamente complejas [Lip87]. Sin embargo, por sí solos, los PMC presentan el inconveniente que debe definirse su estructura en términos de nodos y capas, y en general demoran demasiado para llegar a un mínimo aceptable. Por otra parte, por ser entrenados por el método de gradiente descendente tienen el conocido problema de converger a un mínimo local.

La intención del presente trabajo es demostrar la existencia de una conexión entre los árboles clasificadores y el PMC y como esta relación puede utilizarse para mejorar el desempeño de los AD.

El artículo presentado tiene la siguiente organización. En la sección II se analiza la base de datos IRIS utilizada para las pruebas. Luego, en la sección III, se describe el método utilizado para generar los AD. Posteriormente se explica la forma de obtener un PMC a partir del árbol en la sección IV y la forma de entrenar los PMC en la sección V. Por último los resultados se detallan en sección VI y se exponen las conclusiones en la sección VII.

II. ANÁLISIS DE LA BASE DE DATOS

Esta base de datos fue tomada de [UCI]. La fuente original fue [And35], aunque se hizo famosa a través de los trabajos de R. A. Fisher [Fis36], [Fis50]. Esta base de datos ha sido ampliamente utilizada en la bibliografía y es quizás la mejor conocida en la literatura de reconocimiento de patrones. Para ejemplos referirse a [DuH73], [Das80], [Gat72] y recientemente [AGV95], entre muchos otros.

El conjunto de datos está formado por 150 ejemplos, cada uno de los cuales consta de 4 atributos y una clase. El número de clases es de 3 donde cada clase corresponde a un tipo de planta IRIS. Una de las clases es linealmente separable de las otras dos, estas últimas no son linealmente separables una de otra. En la figura 1 se observa la distribución de las clases en sus componentes principales.

Atributo	Descripción
A_1	Longitud del sépalo en cm
A_2	Ancho del sépalo en cm
A_3	Longitud del pétalo en cm
A_4	Ancho del pétalo en cm

Tabla 1 - Descripción de los atributos.

Clase	Descripción
Clase 0	Iris Setosa
Clase 1	Iris Versicolour
Clase 2	Iris Virginica

Tabla 2 - Descripción de las clases.

Atributo	Valor Mínimo	Valor Máximo	Valor Medio	Desviación Standard
A_1	4.3	7.9	5.84	0.83
A_2	2.0	4.4	3.05	0.43
A_3	1.0	6.9	3.76	1.76
A_4	0.1	2.5	1.20	0.76

Tabla 3 - Estadísticas de los atributos de IRIS

Clase	# ejemplos	(%) del total
Clase 0	50	33.3
Clase 1	50	33.3
Clase 2	50	33.3

Tabla 4 - Distribución de las clases.

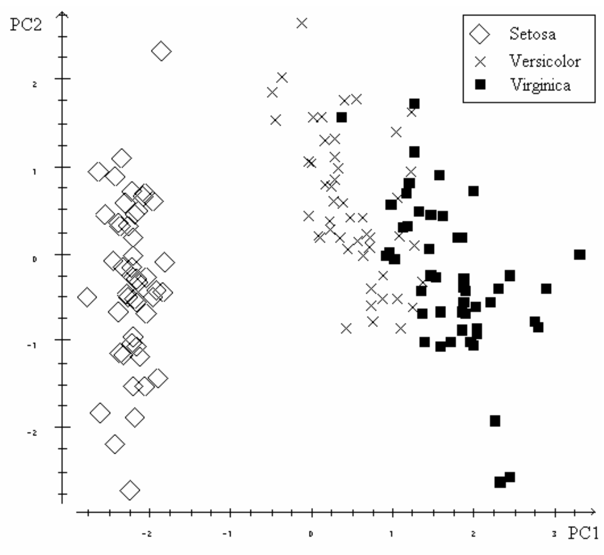


Figura 1- Distribución de las clases según las componentes principales

III. Generación del Árbol de Decisión

El algoritmo utilizado genera AD binarios, es decir, divide el conjunto de ejemplos en cada nodo en dos partes. La selección del atributo para realizar esta partición depende de la información, en términos de la clasificación de los grupos de salida, contenida en cada atributo. El atributo con más información es seleccionado para realizar la partición.

La condición para terminar la generación del árbol es verificar el número de ejemplos en cada nodo; si está por debajo de un valor límite el algoritmo termina. Este mecanismo representa una defensa contra el ruido en los datos.

El árbol generado de esta manera puede verse en la figura 2. Para esto se dividió la base de datos original *IRIS* en dos bases de datos con la mitad de los ejemplos cada una: la primera se utilizó para crear el árbol de decisión y la segunda para realizar la verificación.

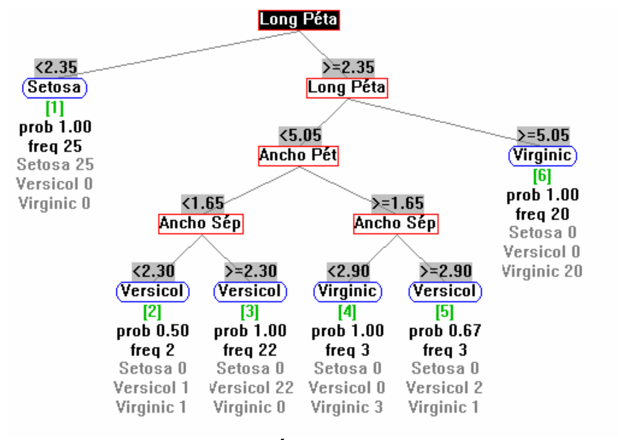


Figura 2 - Árbol de decisión.

Para realizar el podado del árbol se utilizó un algoritmo estadístico, que trabaja desde las hojas hacia arriba removiendo las ramas del árbol que no tienen significado estadístico. Esta significancia es medida usando la prueba de independencia χ^2; y para el trabajo se utilizó un nivel de 0.1%. En la figura 3 se muestra el árbol de decisión una vez podado.

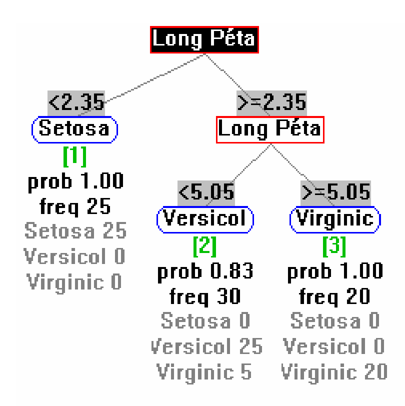


Figura 3 - Árbol de decisión podado.

IV. Mapeo del Árbol de Decisión en un Perceptron Multicapas

Para esta tarea se utilizó un método de mapeo para convertir los árboles de decisión binarios obtenidos en la etapa anterior, en perceptrones multicapas. Implementaciones similares pueden verse en [Set91],[Bre91],[HeF69].

Para entender como un AD puede ser transformado en un PMC de tres capas se debe revisar el procedimiento de clasificación del AD. La clasificación se lleva a cabo atravesando el árbol desde su raíz hasta una de las hojas utilizando el patrón desconocido. La respuesta producida por el patrón es la clase o etiqueta de decisión asociada a la hoja alcanzada. Es obvio que todas las condiciones a lo largo de un camino particular deben satisfacerse para llegar a una hoja determinada. De esta manera cada camino o rama de un AD implementa una operación AND sobre un conjunto de subespacios definidos por los nodos intermedios en el camino. Si uno o más nodos resultan en la misma decisión entonces la operación implicada es una OR. Dado que se podría interpretar que una red tipo PMC utilizada para clasificación implementa en la segunda capa una intersección de los hiperplanos generados en la primer capa y posteriormente una unión de las regiones resultantes en la última capa, existe una relación bastante directa entre ambas aproximaciones.

Las reglas que producen la transformación del AD en un PCM son las siguientes:

a) El número de neuronas en la primera capa oculta de la red equivale al número de nodos internos del árbol de decisión (incluidas la raíz). A esta capa se la denomina *capa de partición*.

b) Todos los nodos terminales u hojas tienen una correspondiente neurona en la segunda capa oculta donde se implementa la operación AND o de intersección. Esta capa es la *capa AND.*

c) El número de neuronas en la capa de salida equivale al número de las distintas clases o acciones. Esta capa implementa la operación OR o de unión y se denomina *capa OR.*

d) La conexión entre las neuronas de la capa de partición y las neuronas de la capa AND implementan la jerarquía del árbol.

Un ejemplo gráfico se muestra en la figura 4, donde se puede percibir la aplicación de cada una de estas reglas.

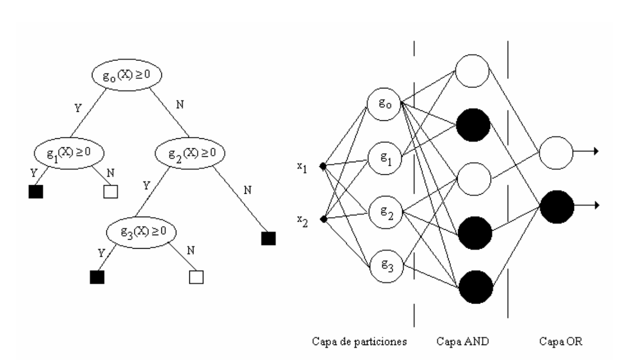


Figura 4 - Procedimiento de mapeo del árbol de decisión en un PMC

Se pueden introducir simplificaciones como no repetir nodos donde se evalúe la misma función de los atributos e incluir la tercer capa solo si hay más de una hoja por clase. En las figuras 5 y 6 se observan los PMC resultantes del mapeo del AD sin podar y podado respectivamente.

V. Entrenamiento del Perceptron

El entrenamiento de los PMC se realizó utilizando el clásico algoritmo de retropropaga-

ción. Esta es una técnica de búsqueda de mínimos por gradiente descendente ampliamente utilizada en este campo [Lip87]. Si bien se podría haber empleado para el entrenamiento una técnica que aprovechara más el análisis realizado por el AD [Set91] se prefirió utilizar retropropagación ya que permite llegar a una representación más flexible.

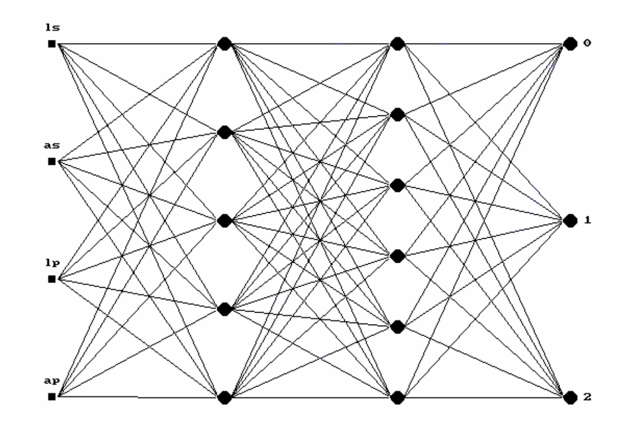


Figura 5 - PMC resultante del mapeo del árbol de decisión sin podar.

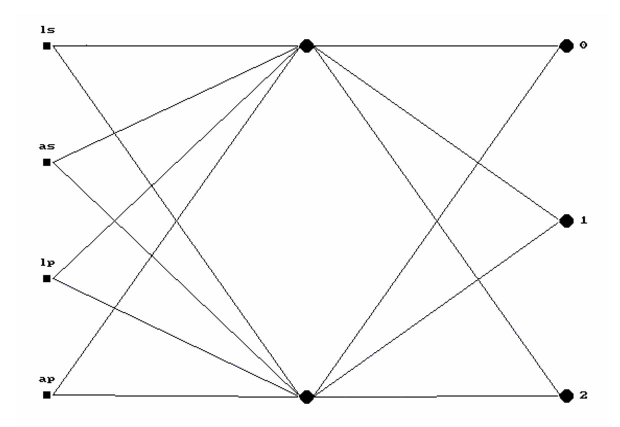


Figura 6 - PMC resultante del mapeo del árbol de decisión podado.

En el caso de los PMC inicializados por el método de mapeo, se comenzó el entrenamiento con un coeficiente de aprendizaje μ=0.5, luego se lo fue decrementando hasta 0.2. El motivo de esto es la suposición de que el error ya se encontraba cerca de un mínimo.

Para los PMC inicializados aleatoriamente se comenzó con un coeficiente μ=1 para luego ir decrementandolo progresivamente.

VI. RESULTADOS

Seguidamente se detallan los resultados de desempeño obtenidos durante el desarrollo de las pruebas. Los desempeños obtenidos por los AD como clasificadores se muestran en la tabla 5.

	Árbol sin podar		Árbol podado	
Clase	Entren.	Prueba	Entren.	Prueba
0	100.0	100.0	100.0	100.0
1	100.0	96.0	100.0	96.0
2	92.0	96.0	80.0	84.0
% total	97.3	97.3	93.3	93.3

Tabla 5 - % de datos clasificados correctamente por los árboles.

Se entrenó un PMC inicializando los pesos de acuerdo al método de mapeo del AD y otro PMC inicializando aleatoriamente los pesos. En el primer caso se entrenó hasta 70 épocas mientras que en el segundo se entrenó durante 210 épocas (Tabla 6). Cabe recordar que cada época consta de 75 ejemplos.

	Inicializac. aleatoria		Inicializac. por mapeo	
Clase	Entrenam.	Prueba	Entrenam.	Prueba
0	100.0	100.0	100.0	100.0
1	100.0	88.0	100.0	96.0
2	100.0	100.0	100.0	96.0
% total	100.0	96.0	100.0	97.3

Tabla 6 - % de datos clasificados correctamente.

En la figura 7 se observa la evolución del error del PMC inicializado por mapeo del AD sin podar y del PMC inicializado aleatoriamente donde se aprecia el mejor comportamiento de la alternativa planteada.

Posteriormente se entrenaron dos PMC en forma similar al caso anterior, solo que se utilizó el AD podado para realizar el mapeo. En el caso del PMC inicializado por mapeo se entrenó hasta 75 épocas mientras que en el otro se entrenó durante 263 épocas. Los resultados se observan en la tabla 7.

Por último se continuó entrenando al PMC con inicialización por mapeo hasta 260 épocas para comparar su desempeño con el otro caso a un mismo número de épocas de entrenamiento. El

desempeño se observa en la tabla 8. Se debe enfatizar que en este caso el PMC superó el desempeño del AD sin podar.

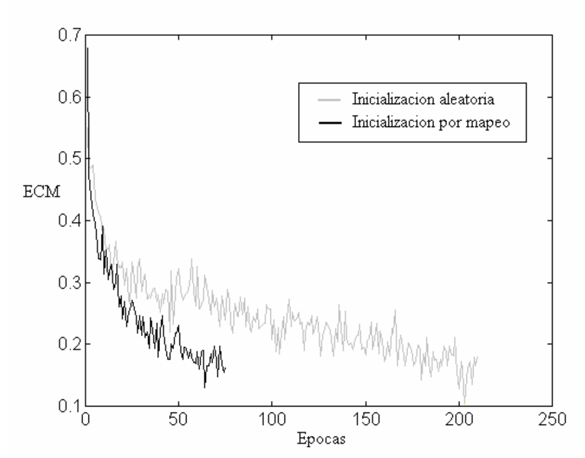


Figura 7 - Evolución del error durante el entrenamiento del PMC

	Inicializac.	aleatoria	Inicializac.	por mapeo
Clase	Entrenam.	Prueba	Entrenam.	Prueba
0	100.0	100.0	100.0	100.0
1	92.0	80.0	92.0	80.0
2	100.0	100.0	100.0	100.0
% total	97.3	93.3	97.3	93.3

Tabla 7 - % de datos clasificados correctamente.

	Inicialización por mapeo	
Clase	Entrenamiento	Prueba
0	100.0	100.0
1	100.0	92.0
2	100.0	100.0
% total	100.0	97.3

Tabla 8 - % de datos clasificados correctamente.

En la figura 8 se muestra la evolución del error para el caso del PMC inicializado por mapeo del AD podado y del PMC inicializado aleatoriamente. Aquí se observa como en el primer caso se parte de un valor menor de error y se converge a un mínimo menor que en el segundo caso.

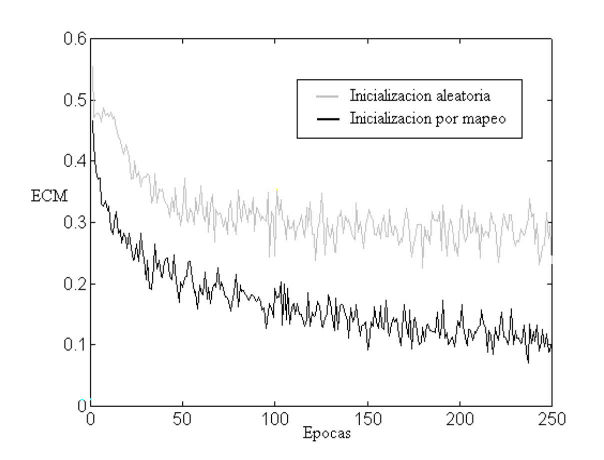


Figura 8 - Evolución del error durante el entrenamiento del PMC

VII. CONCLUSIONES

El método descrito en este trabajo permite definir de manera eficiente la arquitectura de un PMC que actúa como clasificador. Este hecho es de suma importancia ya que no se ha resuelto de manera teórica como definir de manera óptima la arquitectura, y la alternativa de prueba y error consume muchos recursos computacionales. Se podría ver a la generación del AD como un tipo de preproceso sobre los datos que permite utilizar información *a priori* relevante a la estructura de dichos datos para la construcción e inicialización de la red neuronal. El hecho de inicializar los pesos por este método reduce sustancialmente el tiempo de entrenamiento frente a un PMC con la misma arquitectura pero con pesos inicializados aleatoriamente. Por otra parte si se entrena durante suficiente tiempo el desempeño de la red supera al del árbol original con el que fue creada. Esto se debe a la mayor flexibilidad de la estructura para representar relaciones entre los atributos y a su posibilidad de crear regiones arbitrariamente complejas. En un próximo trabajo se analizarán las características de estas regiones de decisión.

Es de destacar la mayor eficiencia lograda por la red creada por mapeo del árbol podado. Esto se puede atribuir a la menor cantidad de parámetros a adaptar, lo que obliga a la red a crear representaciones más compactas que

permiten una mejor generalización a los ejemplos no presentados durante el entrenamiento.

VIII. REFERENCIAS

[AGV95] C. Aviles-Cruz, A.Guérin-Dugué, J.L. Voz, D. Van Cappel, “Enhanced Learning for Evolutive Neural Architecture”, ESPIRIT Basic Research Project Nº 6891, Jun.1995.

[And35] Anderson, E. 1935 “The Irises of the Gaspe Peninsula”, Bulletin of the American Iris Society, 59, 2-5.

[BFO84] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, “Classification and Regression Trees”, Wadsworth. Int. 1984.

[Bre91] R.P. Brent, “Fast Training Algorithms for Multilayer Neural Nets”, IEEE Trans. Neural Networks, vol. 2, nº 3, May 1991.

[Fis36] Fisher,R.A. “The use of multiple measurements in taxonomic problems”, Annual Eugenics, 7, Part II, 179-188,1936.

[Fis50] Fisher, R.A. “Contributions to Mathematical Statistics”, John Wiley, NY, 1950.

[DuH73] Duda, R.O., & Hart, P.E. 1973 “Pattern Classification and Scene Analysis”.(Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. pag. 218.

[Das80] Dasarathy, B.V. “Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments”. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 1, 67-71, 1980.

[Gat72] Gates, G.W. “The Reduced Nearest Neighbor Rule”. IEEE Transactions on Information Theory, May 1972, 431-433.

[GRD91] S.B. Gelfand, C.S. Ravishanker, and E. J. Delp, “An Iteractive growing and pruing algorithm for classification tree design” IEEE Trans. Pattern Anal Machine Intell; Vol. 13 pp. 163 a 164, 1991.

[HeF69] E.G. Henrichon and K.S. Fu, “A Nonparametric Partitioning Procedure for Pattern Classification”, IEEE Trans. Comput., vol. C-18, pp. 614-624, July 1969.

[Lip87] R. P. Lippmann, “An introduction to computing with neural nets” IEEE ASSP Magazing, Apr. 1987.

[LSG91] D. Lee, S.N. Srihari and R. Gaborsky, “Bayesian and Neural Networks Pattern Recognition: A Theorical Connection and Empirical Results with Handwritten Characters”, Art. Neural Nets and Statistical Patern Recognition Old and New Connections, Elsevier Science Publishers, 1991.

[Qui93] J.R. Quinlan, “C4.4”, Morgan Kaufmann 1993.

[SeO90] I. K. Sethi and M. Otten, “Comparison between entropy net and decision tree classifiers,” Proc. Int´l Joint Conf. Neural Networks, Vol.III, San Diego, California, pp.63-68, 1990.

[Set90] I. K. Sethi, “Entropy nets: from decision trees to neural nets” Proceedings of the IEEE, Vol.78, pp. 1605-1613, 1990.

[Set91] I. K. Sethi, “Decision tree performance enhancement using an artificial neural network implementation”, Artificial Neural Networks and Statistical Pattern Recognition Old and New Connections, Elsevier Science Publishers B.V., pp 71-88, 1991.

[UCI] UCI *Repository Of Machine Learning Databases and Domain Theories* (ics.uci.edu: pub/machine-learning-databases).